PARTIAL FILE UPDATING SYSTEM FOR INFORMATION PROCESSOR

Publication number: JP7099674

Publication date: 1995-04-11

Inventor: ISHIKAWA KAZUNARI; SHIBUYA KAZUO; HOSHIAI TAKANARI

Applicant: FUJITSU LTD; NIPPON TELEGRAPH & TELEPHONE

Classification:

- international: ***G06F9/06; G06F9/46; H04M3/00; H04Q3/54;*** **G06F9/06; G06F9/46; H04M3/00; H04Q3/54;** (IPC1-7): H04Q3/54; G06F9/06; G06F9/46; H04M3/00

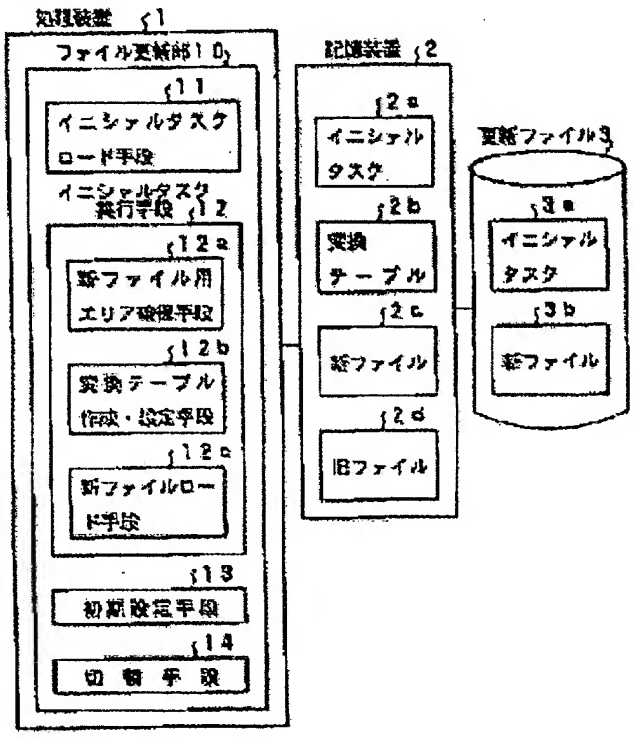
- European:

Application number: JP19930239924 19930927

Priority number(s): JP19930239924 19930927

Abstract of **JP7099674**

PURPOSE:To perform the partial file updating of a part of an application in an online active state by setting a logical/physical address conversion table so as to refer to the same physical memory for data required to be referred to between new and old files in common and enabling the synchronous running of new and old programs.
CONSTITUTION:A processor 1 is provided with a file updating part 10 for updating a part of the old file to the new file. The initial task execution means 12 secures a new area inside a storage device 2 first, then a conversion table 2b for converting logical addresses for the new file to physical addresses is prepared by a conversion table preparation/setting means 12b, also the conversion table for the data taken over from the old file is set in the conversion table 2b and the data in common with the old file are utilized in the new file as well. Then, the new file 2c is loaded to the secured area inside the storage device 2 by a new file loading means 12c.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報(A)　(11)特許出願公開番号

特開平7－99674

(43)公開日　平成７年(1995)４月11日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｈ０４Ｑ	3/54				
Ｇ０６Ｆ	9/06	５４０	Ｆ　9367－5B		
	9/46	３４０	Ｆ　8120－5B		
Ｈ０４Ｍ	3/00		Ｅ　8426－5K		

審査請求　未請求　請求項の数３　ＯＬ　（全 11 頁）

(21)出願番号　特願平5－239924

(22)出願日　平成５年(1993)９月27日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中1015番地
(71)出願人　000004226
日本電信電話株式会社
東京都千代田区内幸町一丁目１番６号
(72)発明者　石川　一成
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内
(72)発明者　渋谷　和夫
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内
(74)代理人　弁理士　穂坂　和雄　（外２名）

最終頁に続く

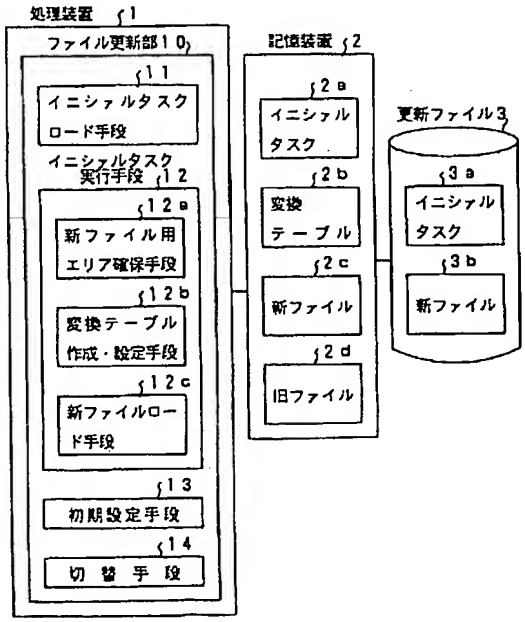
(54)【発明の名称】　情報処理装置の部分ファイル更新方式

(57)【要約】

【目的】本発明はシステム空間に配置されたＯＳとユーザ空間に配置された複数のアプリケーションプログラムが階層構造を備え，各アプリケーション間がメッセージ結合されると共に各アプリケーション内に複数個のタスクが存在可能な構成を備える通信プロセッサ等の情報処理装置における部分ファイル更新方式に関し，オンライン稼働状態のまま，一部のアプリケーションを部分ファイル更新することができることを目的とする。

【構成】処理装置は旧ファイルの一部を新ファイルに更新するファイル更新部を備える。ファイル更新部は，更新対象アプリケーションの旧ファイルの論理空間とは独立の論理空間に新ファイルをロードし，新旧ファイル間で引き継ぐデータについて，同一の物理メモリを参照するよう論理アドレス・物理アドレスの変換を行う変換テーブルを作成設定し，新旧ファイルのプログラムの同時走行を行うよう構成する。

本発明の原理構成図



1

【特許請求の範囲】

【請求項1】　システム空間に配置されたオペレーティングシステム（ＯＳ）とユーザ空間に配置された複数のアプリケーションプログラムが階層構造を備え，各アプリケーション間がメッセージ結合されると共に各アプリケーション内に複数個のタスクが存在可能な構成を備える通信プロセッサ等の情報処理装置において，処理装置は旧ファイルの一部を新ファイルに更新するファイル更新部を備え，ファイル更新部は，更新対象アプリケーションの旧ファイルの論理空間とは独立の論理空間に新ファイルをロードし，新旧ファイル間で引き継ぐデータについて，同一の物理メモリを参照するよう論理アドレス・物理アドレスの変換を行う変換テーブルを作成設定し，新旧ファイルのプログラムの同時走行を行うことを特徴とする情報処理装置の部分ファイル更新方式。

【請求項2】　請求項1において，前記ファイル更新部は，更新ファイルのイニシァルタスクをロードすると起動するイニシァルタスク実行手段を備え，該イニシァルタスク実行手段は，新ファイル用エリアを確保し，論理・物理のアドレス変換を行う変換テーブルを作成して，新ファイルをロードすることを特徴とする情報処理装置の部分ファイル更新方式。

【請求項3】　請求項1において，新・旧ファイルの同時走行状態において，オペレーティングシステムは，ユーザプログラムによるアプリケーションが起動すると，起動アプリケーションを識別して旧ファイルにより処理を行うか，新ファイルにより処理を行うかを判別し，判別結果に応じて対応するファイルにより処理を行うことを特徴とする情報処理装置の部分ファイル更新方式。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】通信プロセッサ等の情報処理装置の部分ファイル更新方式に関する。交換機の制御を行う通信プロセッサ等の情報処理装置では，プログラムやデータを含むファイルを更新する場合，利用者（加入者）に対してサービスを中断しないためにオンライン稼働中に実行する必要がある。このような要求を満たすためには制約があり，その改善が望まれている。

【０００２】

【従来の技術】

〔その1〕：デュプレックス構成のシステムにおけるオンラインファイルの更新方式

図１１は従来のオンライン状態でのファイル更新方式の説明図である。

【０００３】従来技術の一つとして，代表的な部分ファイル更新と言える「パッチ方式」を図１１を用いて説明する。それにより従来技術を２つに分けて記述する。図１１のシステムでは，メモリ及びプロセッサが二重化され，ＭＭ０，ＭＭ１がメモリ，ＰＲ０，ＰＲ１がプロセッサである。

2

【０００４】Ａ．に示す状態ではメモリＭＭ０を用いてプロセッサＰＲ０が稼働（アクト）状態にあり，メモリＭＭ１，ＰＲ１が待機（スタンバイ）状態にあるものとする。各メモリＭＭ０，ＭＭ１にはオペレーティングシステム（ＯＳ）及び各アプリケーションのプログラム（データを含む）が格納されている。この中のアプリケーションプログラムを旧ファイルと呼ぶ。このシステムに対し，アプリケーションプログラム（データを含む）の機能追加等を含むファイル更新を行う場合，片系のメモリＭＭ１に新たなプログラムを含む新ファイルをディスク装置（ＤＫで表示）からロードする。

【０００５】次に，Ｂ．に示すようにロードされた新ファイル側プロセッサＰＲ１で初期設定処理を起動して，メモリＭＭ１の新ファイルによる処理が可能な状態に設定する。この時，旧ファイル側のプロセッサＰＲ０は動作を停止し，新ファイル側プロセッサＰＲ１が初期設定終了後に稼働状態となる。

【０００６】次にＣ．に示すように，プロセッサのアクト・スタンバイの状態を切替えてプロセッサＰＲ１において新ファイルを稼働させ（アクトにし），この稼働状態を監視して正常であれば，旧ファイルのプロセッサＰＲ０のメモリに新ファイルをコピーする。

【０００７】このような，図１１のファイル更新の方式には，次の①，②の２種類がある。

①新／旧ファイル間で，引き継ぎデータがなく，全面的に新ファイルに移行する。この場合は，状態の引き継ぎがないため全て初期設定される。

【０００８】②状態を保持するデータを引き継ぐ。二重化されているメモリの内容は新／旧プロセッサとも同じなので，新ファイルをロードする際に上塗り（前のデータを消去）しないことにより引き継ぐことができる。但し，新／旧ファイル間でデータの構造と割付けが同一であることが条件である。

【０００９】上記した①及び②の方式は，オンライン稼働中のファイル更新でシステム全体を入れ換えていたので，一部のアプリケーションソフト（例えば，特定のサービスを提供するソフトウェアの一部を改良する等）のファイルを更新する場合でも，全体を入れ換えなければならないという問題があった。

【００１０】〔その2〕：パッチ方式

図１４はパッチ方式による部分ファイル更新方式の説明図である。この図１４のシステムでは，シングルプロセッサ構成で，ＭＭがメモリ，ＰＲがプロセッサである。

【００１１】Ａ．に示す状態では，メモリＭＭを用いてプロセッサＰＲがオンライン稼働状態にあるものとする。メモリＭＭにはオペレーティングシステム（ＯＳ）及び各アプリケーションのプログラム（データを含む）が格納されている。この中のアプリケーションプログラムのうち，部分ファイル更新の対象となる一部分を旧ファイルと呼ぶ。また，メモリＭＭには，予め空きエリア

（パッチエリア）が用意されている。

【0012】次にB．に示すようにこのシステムに対して，アプリケーションプログラム（データを含む）の機能追加等を含む部分ファイル更新を行う場合，メモリMMの空きエリア（パッチエリア）に新たなプログラムを含む新ファイルをロードする。この新ファイルは物理アドレスを意識して割り付けられており，空きエリア（パッチエリア）の容量以内のサイズのマシン語で構成される。

【0013】次にC．に示すように旧ファイルの先頭アドレスを新ファイルにジャンプする命令をロードすることにより，旧ファイルから新しいファイルへと切替える。それ以後は，旧ファイルは走行しない。

【0014】

【発明が解決しようとする課題】上記の従来の技術のその1に示したオンライン状態のファイル更新においては，ファイルの一部だけを更新することができないし，システム再開を伴うので，他のアプリケーションへの影響が大きい。また，シングルプロセッサ構成のシステムには適用不可能である。

【0015】また，従来の技術のその2に示したパッチ方式による部分ファイル更新においては，マシン語や物理アドレスにより割り付けやサイズの意識が必要となるので，大量の変更が困難である。さらに，パッチ投入の瞬間に旧ファイルから新ファイルへと切り替わるために，そのアプリケーションが提供するサービスに矛盾が起きてサービスの連続性を損なう可能性がある。この場合，サービス中断の可能性がある。

【0016】本発明はオンライン稼働状態のまま，一部のアプリケーションを部分ファイル更新することができる部分ファイル更新方式を提供することを目的とする。また，本発明は旧ファイルにより実行されるサービスと新ファイルにより実行される新サービスの両方を同時に走行させて，旧サービスでの中断なしでファイルを更新することができる新部分ファイル更新方式を提供することを目的とする。更に，本発明は新部分ファイルのマシン語や物理アドレスを意識しないで，新部分ファイルのサイズや割付け等に対する制約を受けない新部分ファイル更新方式を提供することを目的とする。更に，本発明はシングルプロセッサ構成のシステムに適用可能とすることを目的とする。

【0017】本発明は前記の目的を特定のプログラム構造とアドレス管理機構を備えるシステムにおいて達成するものである。ここで，本発明が適用される対象となるプログラム構造とアドレス管理機構を図12，図13を用いて説明する。

【0018】図12は対象となるプログラム構造の説明図である。(a) に示すように，このプログラム構造は，プログラムとしてシステム空間に割り当てられるものとユーザ空間に割り当てられるものとがあり，システム空

間には基本オペレーションシステム（OSで表示）と拡張OSとに階層化されたOSのプログラムが配置され，ユーザ空間には，複数のアプリケーションに対応した各アプリケーションプログラム（APL1～APLn）が配置されている。このような階層構造を持つOSとしては，例えばCTRONが知られている。この構造では，各アプリケーション（以下，APLiという）は，他のアプリケーションとの間でメッセージにより結合（通信）し，各APLiと拡張OSとの間はシステムコールによるか，メッセージにより結合（通信）する。

【0019】図12の(b) に各アプリケーション（APLi）のプログラム構造を示す。各アプリケーション（APLi）aは，プログラムコード部bと共通データ部cが配置されると共に，起動により生成され，終了により消滅する複数のタスクdがスタック域に存在可能に構成され，プログラムコード部bは，(a) に示すユーザ空間に配置される。

【0020】図13は前提となるアドレス管理機構を示す。上記図12に示すシステム空間とユーザ空間は，それぞれ仮想メモリ上のシステム域Sと各ユーザ域（APLi毎）A，B，C（アプリケーションプログラムが3つの場合）に割当てられ，仮想メモリのシステム域Sと各ユーザ域A～Cの論理アドレスはそれぞれ物理メモリ上の物理アドレスに変換されてアクセスされる。また，仮想メモリ上のシステム域またはユーザ域は，OSの制御レジスタa，制御レジスタbにより指定され，ユーザ域の各APLiは仮想メモリ（論理空間）の切替え（タイムシェアリング）により同時には一つだけ駆動される。

【0021】

【課題を解決するための手段】図1は本発明の原理構成図である。図1において，1は処理装置，2は記憶装置，3は更新ファイルである。処理装置1内の，10はファイル更新部，11はイニシァルタスクロード手段，12はイニシァルタスク実行手段，12aは新ファイル用エリア確保手段，12bは変換テーブル作成・設定手段，12cは新ファイルロード手段，13は初期設定手段，14は切替手段であり，記憶装置2内の2aはイニシァルタスク，2bは新ファイル用の論理アドレスと旧ファイル用の論理アドレスを同一の物理アドレスへ変換する変換テーブル，2cは新ファイル，2dは旧ファイルであり，更新ファイル3内の3aはイニシァルタスク，3bは新ファイルである。

【0022】本発明は旧ファイルが存在する状態で，その中の一部のアプリケーションプログラムの内容を更新する新ファイルを記憶装置の別の論理的なエリアに格納し，この時新旧ファイル間で共通に参照する必要のあるデータについては，同一の物理メモリを参照するように論理・物理アドレス変換テーブルを設定して，新旧プログラムの同時走行を可能にする。

【0023】
【作用】図1において，処理装置1はファイル（旧ファイル）の一部を更新する場合，ファイル更新部10が駆動される。最初にイニシャルタスクロード手段11により，イニシャルタスク及び新たなプログラム（APL）を含む更新ファイル3からイニシャルタスク3aを取り出し記憶装置2にロードする。この後ファイル更新部10はイニシャルタスク実行手段12が起動される。
【0024】イニシャルタスク実行手段12は，最初に新ファイル用エリア確保手段12aにより記憶装置2内に新たなエリアを確保し，次に変換テーブル作成・設定手段12bにより，新ファイル用の論理アドレスを物理アドレスへ変換する変換テーブル2bを作成すると共に変換テーブル2bに旧ファイルからの引き継ぎデータ用の変換テーブルを設定して，旧ファイルと共通のデータを新ファイルでも利用できるようにする。
【0025】次いで，新ファイルロード手段12cにより新ファイル2cを記憶装置2内の上記により確保されたエリアにロードする。ロードされた新ファイル2cは旧ファイル2dとは別の領域に位置するので，旧ファイルと異なるサイズでも問題ない。次に初期設定手段13を起動して新ファイル2cに含まれた初期設定の指示に応じて動作し，他のユーザプログラムのタスクとの間の通信媒体を記憶装置2内に生成する等の動作を行う。次に切替手段14を起動して，拡張OSの管理するエントリー部を旧ファイルから新ファイルへと切替えることで，以後新たに発生するタスクを全て新ファイル用としつつ，以前からの旧ファイル2dと共に新ファイル2cによる動作の両方が走行する状態へ切替えられる。この後，拡張OSはタスクに起動をかける際に，タスクが保持する情報により対応する旧または新ファイルを選択する。また，ファイル更新部10は周期的に残存するタスクのチェックを行い，旧ファイルに対応するタスクの有無を調べ，旧ファイルを必要としなくなった時点でその旧ファイルの論理空間を開放する。
【0026】
【実施例】図2は本発明の実施例の情報処理装置の構成図である。図中，20はプロセッサ（CPU及びメモリを含む）であり，交換機の制御を行う通信用プロセッサ等に相当する。21，22はオペレーティングシステム（OS）であり，21が基本OS，22が拡張OSである。23はアプリケーションプログラム（APL）であり，23aは本発明により設けられたアプリケーションプログラムを更新するためのファイル更新用アプリケーション，23bは更新の対象となる旧アプリケーション（旧APL），23cはその他複数のユーザプログラム（アプリケーションプログラム）である。24はハードディスク（HD），25はハードディスク24内に格納されたファイル更新用ファイル，26はイニシャルタスク，27は更新前のアプリケーションプログラム（旧APL）の内容を変更して新たに作成された新アプリケーションプログラム（新APL），27aは初期設定ルーチン，27bはユーザに使用されるアプリケーションプログラムの本体（APL）である。
【0027】この図2の情報処理装置は上記の図12及び図13に示すようなプログラム構造とアドレス管理機構を備え，基本OS，拡張OS及びアプリケーションプログラム間の階層構造と相互関係についても上記図12及び図13について説明したものと同様である。
【0028】図3乃至図5はファイル更新における各部の処理シーケンス（その1）～（その3）である。図2の構成における図3乃至図5に示すファイル更新のための各部（基本OS，拡張OS，アプリケーション23）の処理シーケンスを，図6乃至図8に示す各処理における状態図（その1）～（その3）を参照しながら説明する。なお，図6乃至図8の各状態を示す図では，図2と異なりメモリとプロセッサとを分離して示した。
【0029】図3において，アプリケーション（APL）23において，ファイル更新用APL23aが動作して，ハードディスク（HD）にファイル更新用ファイル25をロードする要求を基本OS21に対し行う（図3のa）。これにより基本OS21はファイル（Fで表示）更新用新ファイル25を図示されない外部記憶装置からハードディスク24にロードする（図3のb）。この状態を図6の(1)に示す。
【0030】次にアプリケーション23はイニシャルタスク引き上げを拡張OS22に対し要求すると（図3のc），拡張OS22はハードディスク24からイニシャルタスク（図2の26）をメモリへ引き上げる（図3のd）。この状態を図6の(2)に示す。続いてアプリケーション23はイニシャルタスクを起動すると，イニシャルタスクが実行されて新APL用エリアの確保を要求する（図3のe）。これにより基本OS21はエリア確保を行う（図3のf）。この状態を図6の(3)に示す。
【0031】アプリケーション23は次に，論理アドレス変換テーブルの作成を指示し（図3のg），これに応じて基本OSで論理アドレス変換テーブルが作成され（同g'）。続いてアプリケーション23は続いて引き続きデータ（旧APLで使用したデータを新APLでも使用する共通データ）の論理アドレス変換テーブルの設定指示をすると（同h），基本OSで論理アドレス変換テーブルが設定される（同h'）。この状態は図6の(4)に示され，旧APLとは異なる論理変換テーブルが設定される。続いて，アプリケーション23は新ファイルのロードを要求する（同i）。基本OS21はこれに応じて，ハードディスク内の新ファイル（図2の新APL27）をメモリにロードする（図4のj）。この状態は図7の(5)に示され，新APLが既に確保されたエリアに格納される。
【0032】ここで，上記論理アドレス変換テーブルの

作成，設定，及び新ファイルのロードの処理（図3のg～図4のj）により発生する新APLと旧APLのメモリ領域の関係を図9を用いて説明する。

【0033】図9は本発明によるアドレス管理機構の説明図である。図に示すように仮想メモリ上に旧APL空間91が配置され，これと異なる位置に新APL空間92が割り当てられ，各APL空間内には，それぞれ新・旧の共通データ，コード（プログラムコード），及びスタック（タスクが格納される領域）が格納されている。この仮想メモリ上の新・旧のAPL空間92，91は，OSに設けられた制御レジスタ90により，仮想メモリ上のアドレスである論理アドレス空間を切替えることによりアクセスされる。

【0034】この論理アドレスは，論理アドレス変換テーブルにより物理メモリ95上の位置を表す物理アドレスへ変換されるが，旧APL空間のアドレスは旧APL用の論理アドレス変換テーブル93（点線により変換機能を表す）により物理メモリ95上の各領域a，b，cのアドレスへ変換される。また，新APL用の論理アドレス変換テーブル94（同じく点線で変換機能を表す）は，共通データ（引き継ぎデータ）については物理メモリ95上の旧APLの共通データの領域aのアドレスに変換するよう設定され，コード及びスタックの論理アドレスについては新APL用の領域d，eの物理アドレスに変換するよう設定されている。このように，新APLの論理アドレス変換テーブルを論理アドレス変換テーブルの作成及び設定の処理（図3のg，g'，h及びh'）において設定することにより引き継ぎデータが新APLで利用することができる。

【0035】図4の説明に戻ると，新ファイルのロード（図4のj）の後，アプリケーション23はメモリにロードされた新ファイルの初期設定ルーチン（図2の27aに対応）を起動する（図4のk，図7の(6)）。この初期設定ルーチンで他のユーザプログラムとのタスク間通信媒体を生成する要求を行うと，拡張OSがこれに応じてタスク間通信媒体（タスク間で送受信されるメッセージの格納領域，メッセージボックス，メッセージID等）を生成する（図4のk，l）。また初期設定ルーチンにより新APL用の作業データ等の初期設定が行われ（図4のm），拡張OSのエントリー部が旧APLのものに切替わり（同m'），以後新たに生成されるタスクは新APLが対象となり，ファイル更新を終了する。図7の(7)にファイル更新終了後の状態を示す。

【0036】この後，新／旧アプリケーションが同時走行する状態に移行し，この状態では旧ファイルにより処理を行うタスクが存在するか否かを拡張OSに対し問い合わせて監視する（図4のn）。この状態を図7の(8)に示す。

【0037】すなわち，更新が行われたアプリケーションを起動した場合，拡張OSにより，そのアプリケーションが新ファイル（アプリケーションプログラム）により処理すべきアプリケーションか，旧ファイル（アプリケーションプログラム）により処理すべきアプリケーションであるかを識別して（図4のo），識別結果に応じてアプリケーションに対し新APLで処理するか旧APLで処理するか振り分けて，対応するアプリケーションが実行され（図4のp，q），各アプリケーションにより要求された処理は拡張OSから起動実行されて，アプリケーションの処理が終了する。

【0038】ここで，新ファイル／旧ファイルが共存する状態におけるメッセージ振り分けの処理を説明する。

イ．タスク使用形態

タスク使用形態を分類すると，図10に示すように2つのタイプがある。

【0039】タイプ1：図10のA．に示すようにシステム初期設定時に生成され，以降システム運転中に存続し続けるタスクである。

タイプ2：図10のB．に示すように初期設定時に入口タスクを生成し，以後システム運転中に動的に生成・消滅するタスクであり，システム運転中はメッセージにより入口タスクから生成されるタスクである。

ロ．タスク間通信パターン

パターンa：メッセージ送信側がタイプ1でメッセージ受信側がタイプ1

パターンb：メッセージ送信側がタイプ2でメッセージ受信側がタイプ1

パターンc：メッセージ送信側がタイプ1でメッセージ受信側がタイプ2

パターンd：メッセージ送信側がタイプ2でメッセージ受信側がタイプ2

ハ．ファイル更新中のメッセージの振り分け処理

①特定タスクへのメッセージ送信時の処理

(1) パターンa，bの時（受信側がタイプ1の時）

タイプ1のタスクには，内部状態を持たない（管理情報はメモリの共通域に置く）という条件を付けることにより，即時に全メッセージを新タスク（タイプ1）に送信してもよい。この条件を満たすためタイプ1のタスクでメッセージ受信を行う場合，そのプログラムは先頭でメッセージ待ち（Wait）になるようにする。これによりローカル変数（旧ファイルを使用する特定タスク内で保持する変数）の使用を回避する。また，タイプ1のタスクは，例えばウエイト状態にあったとしても，ファイル更新開始時点で削除しても構わない。

【0040】(2) パターンc，dの時（受信側がタイプ2の時）

タイプ2のタスクの使用形態を考慮すると，同一ユーザ付与名称を有するタイプ2のタスクが新旧同時に存在することはあり得ない。従って，メッセージ送信先のタスクは一意に定まる。

【0041】②タイプ2におけるタスク生成処理

常に新ファイルでタスク生成を行う（新ファイルの入口タスクに送信する）。

ニ．上記ハ．の機能の実現方法

①ユーザ付与タスク名称に対応するタスク識別番号（ＩＤという）との対応関係を管理する。すなわち，同一ユーザ付与名称に対応するタスクＩＤが複数存在することを許容し，それぞれのタスクの新旧を管理する。

【００４２】②タスクタイプとタスクＩＤの対応関係を管理する。

③メッセージ通信時の新旧タスクへの振り分け処理を行う。

具体例により説明すると，通信プロセッサの場合，個々の呼（発呼，着呼）に対してサービス（接続処理，通話状態の監視，接続解放等）を行うタスクはタイプ１，タイプ２の何れでもよいが，呼が発生した時に旧ＡＰＬによりタスクが生成した後，ファイル更新により新ＡＰＬが起動しても，旧ＡＰＬにより生成したタスクはそのタスクが消滅するまで旧ファイルにより動作を継続する。新ＡＰＬが起動した後に，発生するタスク呼に対しては新ファイルによるタスクが生成して消滅するまで新ファイルにより動作する。

【００４３】図４に続いて図５のシーケンスの説明をすると，新旧ＡＰＬが同時走行した後，オペレーティングシステム（基本ＯＳ，拡張ＯＳ）及びアプリケーションにおいて一定時間監視が行われる（図５のｒ）。すなわち，ファイル更新アプリケーション（ＡＰＬ）によりファイル更新後処理が開始され，旧ＡＰＬのタスクの存在の有無を拡張ＯＳに対し問い合わせる（図５のｓ）。拡張ＯＳはその問い合わせに対し，管理情報を用いて旧ＡＰＬに対応するタスクの有無を識別し（同ｔ），アプリケーションへ通知する。この場合，旧ＡＰＬに対応するタスクが依然として有る場合は，再びファイル更新ＡＰＬのファイル更新後処理（図５のｓ）に戻り，一定時間毎の問い合わせ（拡張ＯＳへ）を行う。

【００４４】旧ＡＰＬが無くなった場合は，旧ＡＰＬ切離しの指示を行い（同ｕ），旧ＡＰＬのエリアを解放する指示を発生する（同ｖ）。これに応じて基本ＯＳは旧ＡＰＬのエリアを解放する処理を行う（同ｗ）。この状態を図８の(9) に示す。

【００４５】エリアが解放されると，基本ＯＳではガーベジコレクションが実行される（同ｘ）。このガーベジコレクションは，メモリ上に散在する解放されて使用されない空きエリアを統合する処理であり，この様子を図８の(10)に示す。このように，旧ＡＰＬが存続する間は上記のファイル更新後処理（図５のｓ）による監視が継続して行われ，終了すると旧ＡＰＬは消滅する。

【００４６】上記の図３乃至図５の処理シーケンスは，二重化されないシステムにおいて，通常のサービス処理を実行しながらオンライン状態で実行されるが，二重化されたシステムにおいてアクト系で実行できることは当然である。

【００４７】

【発明の効果】本発明によればオンライン状態のファイル更新で，一部のアプリケーションだけ部分ファイル更新を実現することができる。また旧アプリケーション，新アプリケーションの両方を走行させるので，サービスの中断無しにファイル更新が可能となる。更に，論理アドレス空間を用いるので物理アドレスを意識しなくても良いため新アプリケーションとして用意する新ファイルに対して，サイズや割付けの制約がない。また，ソースモジュールレベルで修正ができることにより，大量の変更が可能となる。更に，部分ファイル更新にシステム再開を伴わないので，他の処理に対する影響が小さく抑えられる。また，本発明はシングルプロセッサシステムにも適用が可能である。

【図面の簡単な説明】

【図１】本発明の原理構成図である。

【図２】本発明の実施例の情報処理装置の構成図である。

【図３】ファイル更新における各部の処理シーケンス（その１）である。

【図４】ファイル更新における各部の処理シーケンス（その２）である。

【図５】ファイル更新における各部の処理シーケンス（その３）である。

【図６】各処理における状態図（その１）である。

【図７】各処理における状態図（その２）である。

【図８】各処理における状態図（その３）である。

【図９】本発明によるアドレス管理機構の説明図である。

【図１０】タスク使用形態の説明図である。

【図１１】従来のオンライン状態でのファイル更新方式の説明図である。

【図１２】対象となるプログラム構造の説明図である。

【図１３】前提となるアドレス管理機構である。

【図１４】従来のパッチによる部分ファイル更新方式の説明図である。

【符号の説明】

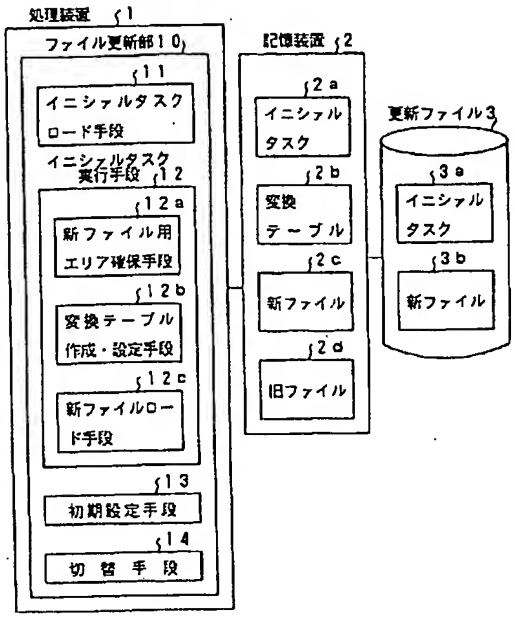
１　処理装置
１０　ファイル更新部
１１　イニシァルタスクロード手段
１２　イニシァルタスク実行手段
１２ａ　新ファイル用エリア確保手段
１２ｂ　変換テーブル作成・設定手段
１２ｃ　新ファイルロード手段
１３　初期設定手段
１４　切替手段
２　記憶装置
２ａ　イニシァルタスク
２ｂ　変換テーブル

2c　新ファイル
2d　旧ファイル
3　更新ファイル

3a　イニシャルタスク
3b　新ファイル

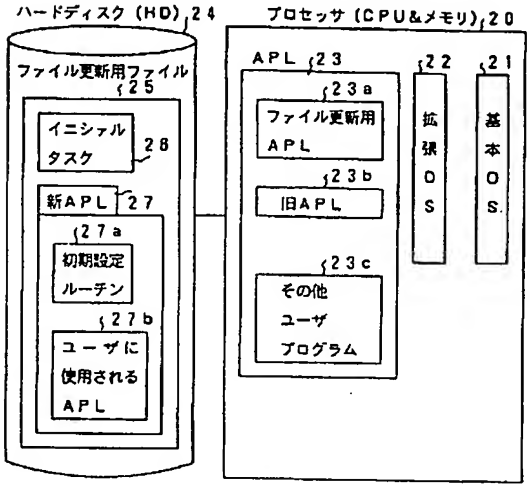
【図1】

本発明の原理構成図



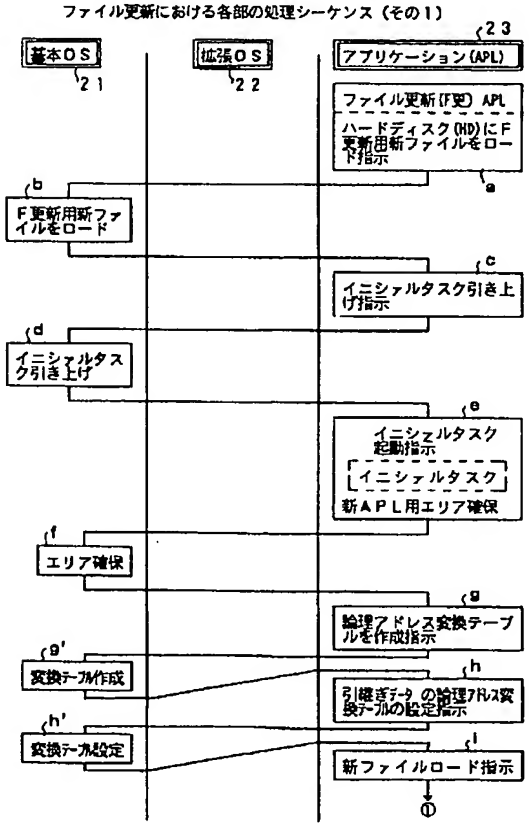
【図2】

本発明の実施例の情報処理装置の構成図



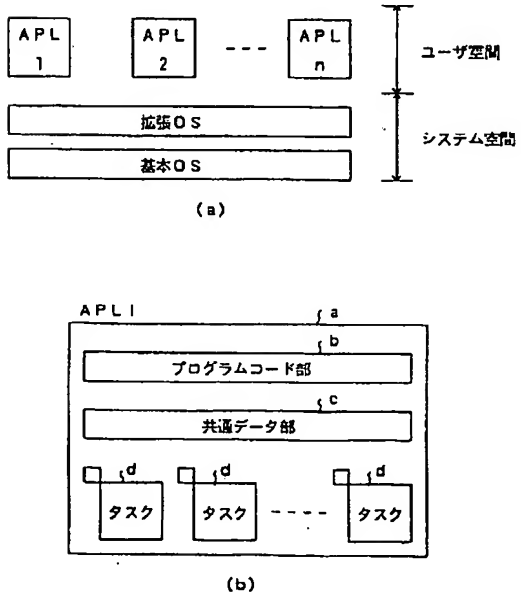
【図3】

ファイル更新における各部の処理シーケンス（その1）



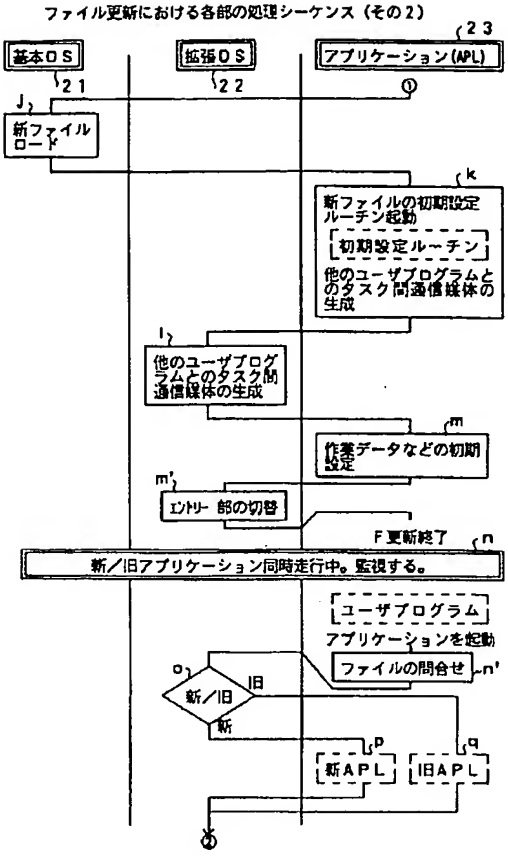
【図12】

対象となるプログラム構造の説明図



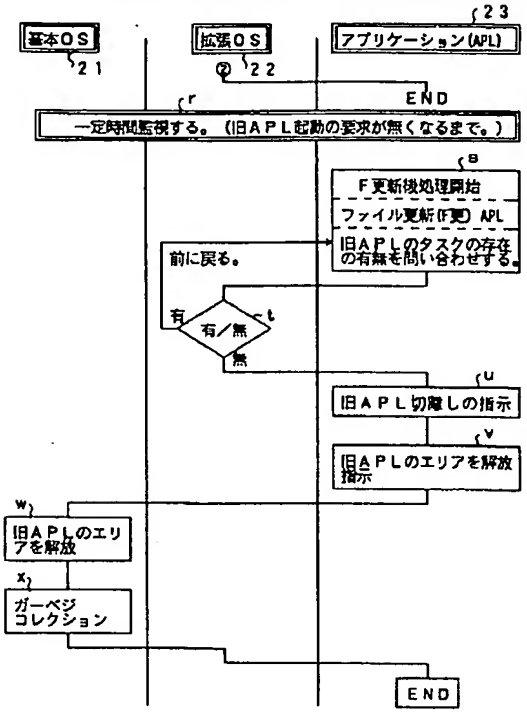
【図4】

ファイル更新における各部の処理シーケンス（その2）



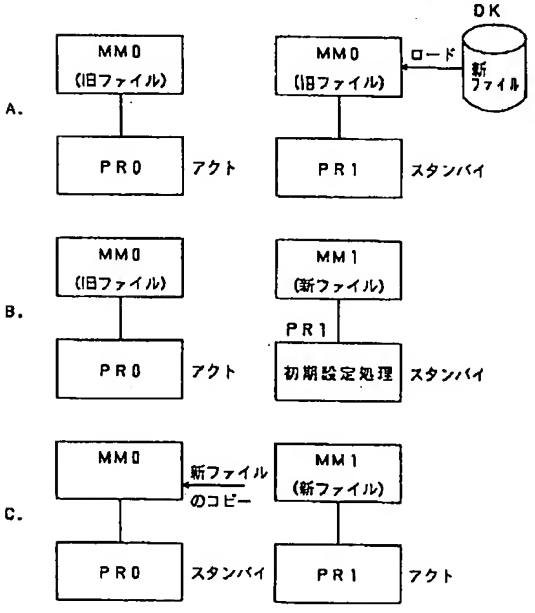
【図5】

ファイル更新における各部の処理シーケンス（その3）

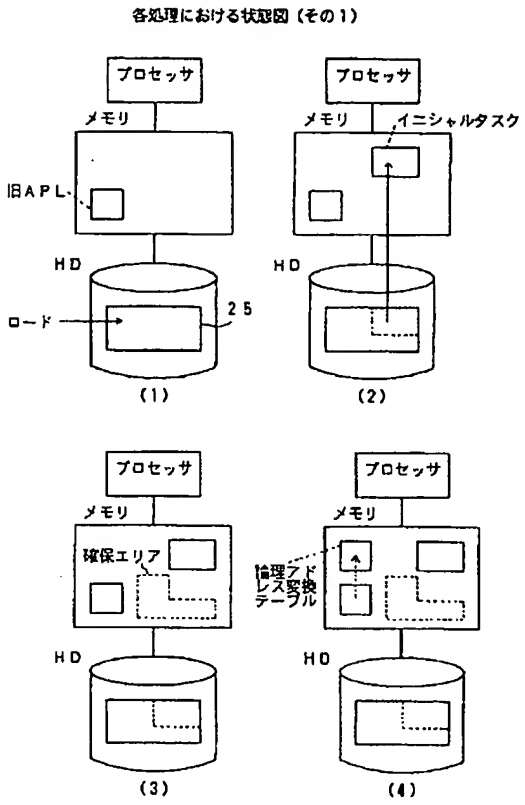


【図11】

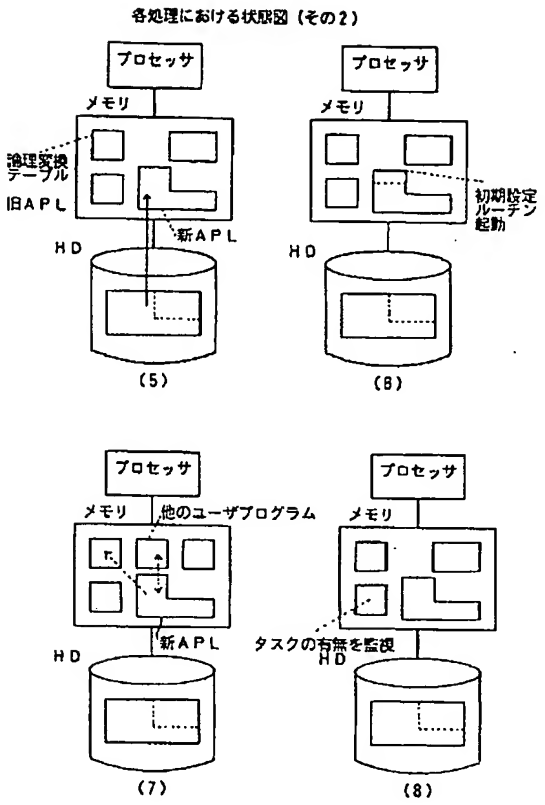
従来のオンライン状態でのファイル更新方式の説明図



【図６】

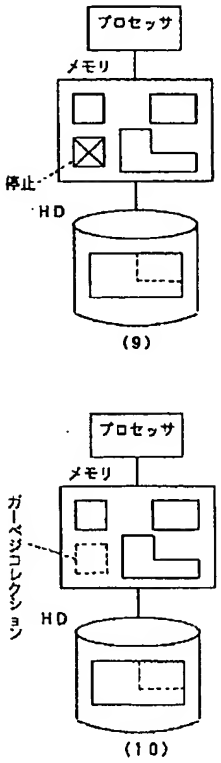

各処理における状態図（その１）
プロセッサ
メモリ
旧ＡＰＬ
ＨＤ
ロード
２５
(1)
プロセッサ
メモリ
イニシャルタスク
ＨＤ
(2)
プロセッサ
メモリ
確保エリア
ＨＤ
(3)
プロセッサ
メモリ
論理アドレス変換テーブル
ＨＤ
(4)

【図７】


各処理における状態図（その２）
プロセッサ
メモリ
論理変換テーブル
旧ＡＰＬ
新ＡＰＬ
ＨＤ
(5)
プロセッサ
メモリ
初期設定ルーチン起動
ＨＤ
(6)
プロセッサ
メモリ
他のユーザプログラム
新ＡＰＬ
ＨＤ
(7)
プロセッサ
メモリ
タスクの有無を監視
ＨＤ
(8)

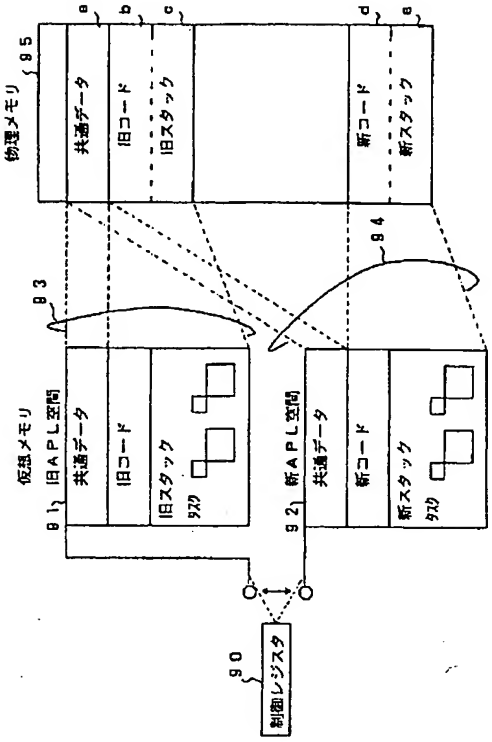
【図8】

各処理における状態図（その3）



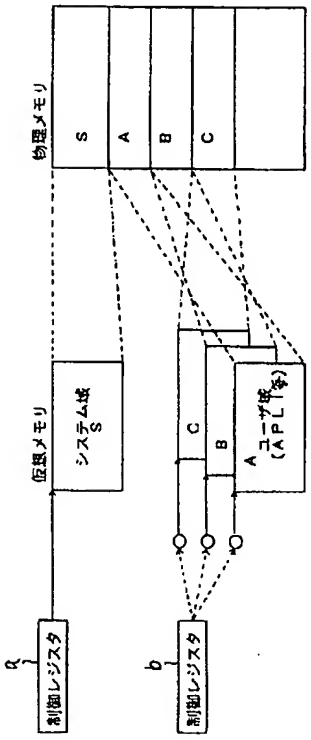
【図9】

本発明によるアドレス管理機構の説明図



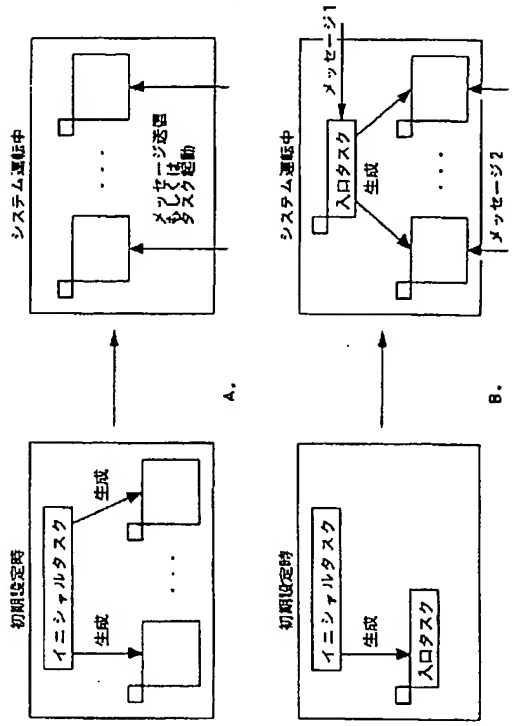
【図13】

前提となるアドレス管理機構



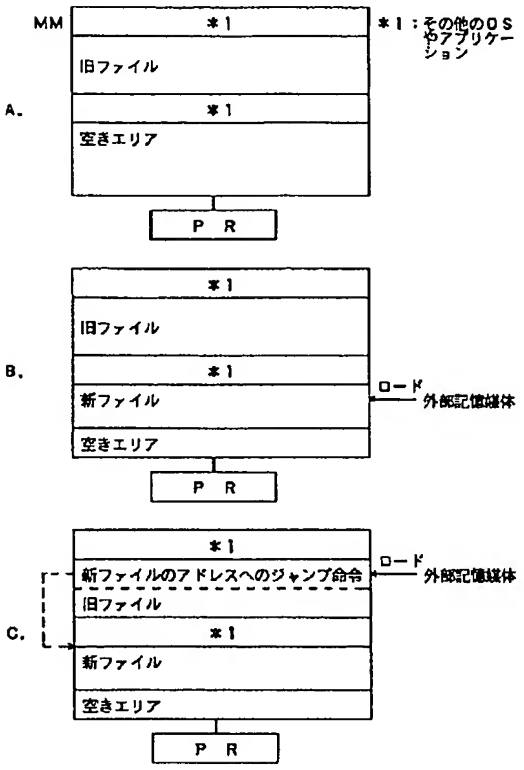
【図１０】

タスク使用形態の説明図



【図１４】

従来のパッチによる部分ファイル更新方式の説明図



フロントページの続き

(72)発明者　星合　隆成
東京都千代田区内幸町１丁目１番６号　日本電信電話株式会社内